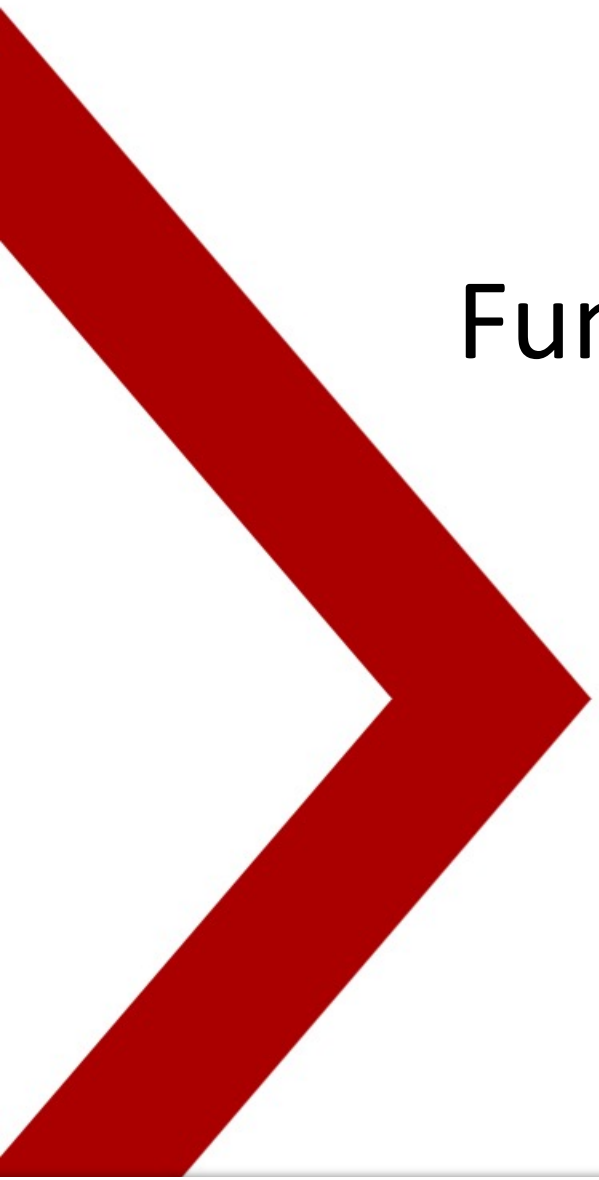


# Function

# Topics



1. Introduction to function
2. Defining a function
3. Parameter and variable scopes
4. Return statement
5. Tips

# 1) INTRODUCTION TO FUNCTION

# What is function?



- Function is a sub-program making your code shorter and better understanding.
- 1) Built-in functions
  - print("hello world")
  - x = round(2/3, 2) # 0.67
- 2) User-defined function
  - seq_sum(a,b)

```
1 def seq_sum(a, b):
2   sum = 0
3   for i in range(a, b+1):
4     sum += i
5   return sum
```

# Why function?

## 1) Shorter code, better understanding



- Calculate a summation from 1 to n

```
n1 = 10
sum1 = 0
for i in range(1, n1+1):
  sum1 += i
print(sum1)

n2 = 100
sum2 = 0
for i in range(1, n2+1):
  sum2 += i
print(sum2)

n3 = 200
sum3 = 0
for i in range(1, n3+1):
  sum3 += i
print(sum3)
```

```
55
5050
20100
```

```
def sum_seq(n):
  sum = 0
  for i in range(1, n+1):
    sum += i
  return sum

sum1 = sum_seq(10)
sum2 = sum_seq(100)
sum3 = sum_seq(200)

print(sum1)
print(sum2)
print(sum3)
```

```
55
5050
20100
```

# Why function? (cont.)

## 2) Update only one place, leading less errors



- Update to calculate a summation from a to b

```
1 a1 = 5; b1 = 10
2 sum1 = 0
3 for i in range(a1, b1+1):
4   sum1 += i
5 print(sum1)
6
7 a2 = 10; b2 = 100
8 sum2 = 0
9 for i in range(a2, b2+1):
10   sum2 += i
11 print(sum2)
12
13 a3 = 100; b3 = 200
14 sum3 = 0
15 for i in range(a3, b3+1):
16   sum3 += i
17 print(sum3)
```

```
45
5005
15150
```

```
1 def sum_seq(a, b):
2   sum = 0
3   for i in range(a, b+1):
4     sum += i
5   return sum
6
7 sum1 = sum_seq(5, 10)
8 sum2 = sum_seq(10, 100)
9 sum3 = sum_seq(100, 200)
10
11 print(sum1)
12 print(sum2)
13 print(sum3)
```

```
45
5005
15150
```

# How to use function



- What to know?
  - 1) Function name
  - 2) Input (parameters)
  - 3) Output (return value)

- x = abs(-5) # 5
- y = round(2/3, 2) # 0.67

- print("hello world")

# Start with a simple function



1) Function name
2) Input (parameters)
3) Output (return value)

0) define
1) start
2) call fn

```
1 def hello(name):
2   print("Hello", name)
3
4 BLACKPINK = ["Jisoo", "Jennie", "Rosé", "Lisa"]
5 for e in BLACKPINK:
6   hello( e )
```

```
Hello Jisoo
Hello Jennie
Hello Rosé
Hello Lisa
```

# Start with a simple function (cont.)

## Function can call other functions.


1) Function name
2) Input (parameters)
3) Output (return value)

0) define

0) define

1) start

2) call fn

```
1 def hello(name):
2   print("Hello", name)
3
4 def hello_all(names):
5   for e in names:
6     hello(e)
7
8 BLACKPINK = ["Jisoo", "Jennie", "Rosé", "Lisa"]
9 hello_all(BLACKPINK)
```

```
Hello Jisoo
Hello Jennie
Hello Rosé
Hello Lisa
```



python powered

# Exercise



- Ex1: Happy birthday to you
- Ex2: HDB(name)

# 2) DEFINING FUNCTION

# Components of User-Defined Function


1) Function name
2) Input (parameters)
3) Output (return value)

```
1 def dot( u, v ) :
2     p = 0
3     for i in range(len(u)):
4         p += u[i]*v[i]
5     return p
6
7 u = [1, 2, 3]
8 v = [2, 2, 2]
9 w = dot( u, v )
10 print(w)
```

```
12
```

# Exercise (return single value)



- Ex3: Farenheit to celsius
- Ex4: Compute a triangle area
- Ex5: Prime number

# 3) PARAMETERS AND VARIABLE SCOPES

# Input parameters

```
1 def sum_seq(n):
2   sum = 0
3   for i in range(1, n+1):
4     sum += i
5   return sum
6 #----------------------------------
7 def add_one(n):
8   return n+1
9 #----------------------------------
10 def is_even(n):
11   return ((n % 2) == 0)
12 #----------------------------------
13
14 n = 10
15 out1 = sum_seq(n)
16 out2 = add_one(n)
17 out3 = is_even(n)
```

```
55
11
True
```

- Can be duplicated with other functions
- Cannot be used by other functions.

# Input parameters (cont.)

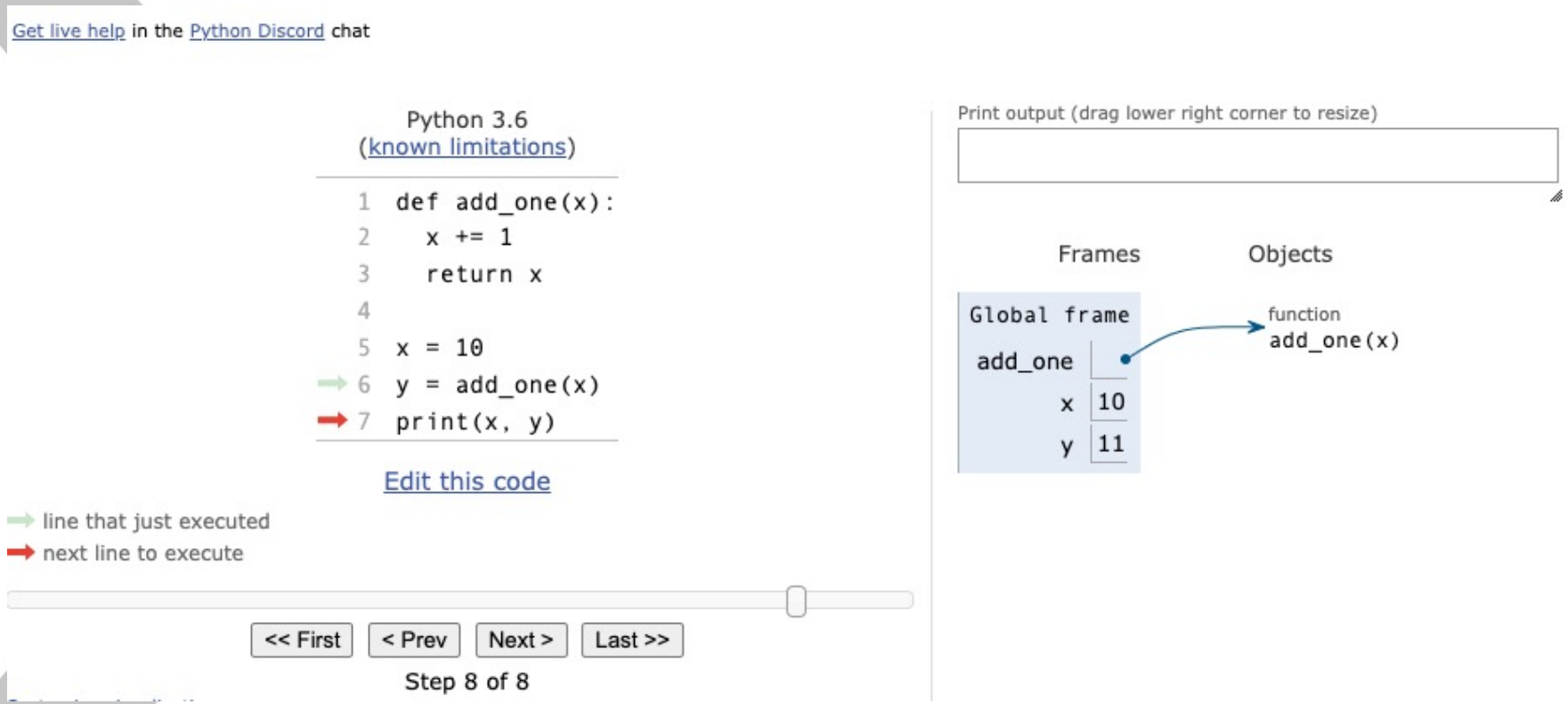


```
1 def distance(x1, y1, x2, y2):
2     dx = x1 - x2
3     dy = y1 - y2
4     d = (dx**2 + dy**2)**0.5
5     return d
6
7 d1 = distance(10.5, 12.0, 30.0, 50.5)
8 x = 10.5; y = 11.2
9 d2 = distance(x, y, 30.0, 50.5)
```

```
d1 = 43.15669125408017
d2 = 43.871858861917396
```

# Variable Scope: Local variables





# Variable Scope: Global variables



# Exercise (return list)



- Ex6: Add two lists
- Ex7: Convert from list of number to list of string
- Ex8: Get positive elements

# 4) RETURN

# Return usages

- 1) just end and no return values
- 2) end and return a single value
- 3) end and return multiple values

# Return (1):
# just end and no return value



```
1 def hello(name):
2   print("Hello", name)
3   return
4
5 def hello(name):
6   print("Hello", name)
7
8 def hello(name):
9   if name == "":
10    return
11  print("Hello", name)
```

```
15 hello("Pop")
16 s = hello("Pop")
17 print(s)
```

```
Hello Pop
Hello Pop
None
```

# Return (2):
# End and return a single value



```
1 def read_vector():
2   x = input().split()
3   v = []
4   for i in range(len(x)) :
5     v.append( float(x[i]) )
6   return v
7
8 def dot(u, v):
9   p = 0
10  for i in range(len(u)):
11    p += u[i]*v[i]
12  return p
13
14 u = read_vector()
15 v = read_vector()
16 dot_p = dot(u, v)
17 print("u.v =", dot_p)
```

```
1 2 3 4 5
2 2 2 2 2
u.v = 30.0
```

# Return (3): End and return multiple values



## Tuple (multiple values)

```
1 def roots(a, b, c):
2     t = (b**2 - 4*a*c)**0.5
3     r1 = (-b + t)/(2*a)
4     r2 = (-b - t)/(2*a)
5     return r1,r2
6
7 x1,x2 = roots(6, -6, -36)
8 print(x1,x2)
```

```
3.0 -2.0
```

## List

```
1 def get_odds( x ):
2     odds = []
3     for e in x:
4         if e%2 == 1:
5             odds.append(e)
6     return odds
7
8 x3 = get_odds([1, 7, 2, 10, 5])
9 print(x3)
```

```
[1, 7, 5]
```

https://pythontutor.com/

# Return values via input parameters


Create new list

```
1 def abs_all( data ):
2   d = [0] * len(data)
3   for k in range(len(data)):
4     d[k] = abs(data[k])
5   return d
6
7 x = [1,-2,3,-4,-5]
8 y = abs_all(x)
9 print( 'x =', x )
10 print( 'y =', y )
```

```
x = [1, -2, 3, -4, -5]
y = [1, 2, 3, 4, 5]
```

NOT create new list

```
1 def abs_all( data ):
2   for k in range(len(data)):
3     data[k] = abs(data[k])
4
5 x = [1,-2,3,-4,-5]
6 abs_all(x)
7 print( x )
```

```
[1, 2, 3, 4, 5]
```

# Exercise (update input values)



- Ex9: Fill list values
- Ex10: Clip boundary
- Ex11: Replace values

# 5) TIPS

# Common mistake1: Don't define a variable duplicated to the input parameter



```
1 def f1(x):
2   # doesn't use the input parameter (value)
3   x = 0
4
5 def f2(x):
6   # doesn't use the input parameter (list)
7   x = [0]*len(x)
8
9 a = 9
10 f1(a) # 'a' doesn't change
11 b = [1,2,3,4]
12 f2(b) # 'b' doesn't change
```

# Common mistake2: call function properly

100th Anniversary of Chula Engineering 2013

## Incorrect

```
1 def hello(name):
2   print("Hello", name)
3
4 def double(u):
5   return 2*u
6
7 x = hello("Bell")
8 print('x =', x )
9 double( [1,2,3] )
```

1

2

```
Hello Bell
x = None
[1, 2, 3, 1, 2, 3]
```

## Correct

```
1 def hello(name):
2   print("Hello", name)
3
4 def double(u):
5   return 2*u
6
7 hello("Bell")
8 y = double( [1,2,3] )
9 print(y)
```

1

2

```
Hello Bell
[1, 2, 3, 1, 2, 3]
```

python powered

# Common mistake3: use return statement properly



```
1 def clip(x):
2   if x < 0:
3     return 0
4     x += 2 # action after return
5   return x
6
7 # return only some cases
8 def is_odd(x):
9   if x%2 == 1:
10    return True
```

# Tips



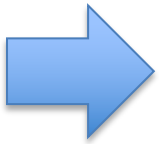
1

```
def is_odd(n):
    if n%2 == 1:
        return True
    else:
        return False
```

```
def is_odd(n):
    return n%2 == 1
```

2

```
def foo(n):
    if n%2 == 1:
        return 3*n+1
    else:
        return n//2
```



```
def foo(n):
    if n%2 == 1:
        return 3*n+1
    return n//2
```

# Conclusion

- Introduction to function
  - Sub-program: (1) built-in and (2) user-defined functions
- Defining a function
  - (1) function name, (2) input parameters, (3) return values
- Parameter and variable scopes
  - Local & global scopes
- Return statement
  - Return1: just end and no return values
  - Return2: end and return one value
  - Return3: end and return multiple values
  - For *list* input, you can decide to update it directly.
- Tips
  - Don't replace the input parameters
  - Call function correctly
  - Return correctly
  - How to write a nice function